What is claimed is:

1. A system for providing network security by managing and manipulating formed data connections and connection attempts initiated over a data-packet-network between at least two nodes connected to the network comprising:

a system host machine connected to the network;

a first software application residing on the host machine for detecting and monitoring the connections and connection attempts;

a data store for storing data about the connections and connection attempts; and

a second software application for emulating one or more end nodes of the connections or connection attempts;

characterized in that the system using the detection software detects one or more pre-defined states associated with a particular formed connection or connection attempt in progress including those associated with any data content or type transferred there over and performs at least one packet generation and insertion action triggered by the detected state or states, the packet or packets emulating one or more end nodes of the connection or connection attempt to cause preemption or resolution of the detected state or states.

2. The system of claim 1 wherein the data-packet-network encompasses a Local Area Network connected to the Internet network enhanced with Transfer Control Protocol over Internet Protocol and User Datagram Protocol over Internet Protocol.

3. The system of claim 1 wherein the system host machine is one of a desktop computer, a router, an embedded system, a laptop computer, or a server.

4. The system of claim 1 wherein the system host is an especially dedicated piece of hardware.

5. The system of claim 1 wherein emulation of the end nodes of the connections or connection attempts is performed by generation and insertion into a data stream of the connection or connection attempt data packets using Transfer Control Protocol over Internet Protocol, the packets emulating packets from the current sending node in the connection.

6. The system of claim 5 wherein the packets inserted into a connection or connection attempt are one or a combination of Transfer Control Protocol reset packets or Transfer Control Protocol FIN packets.

7. The system of claim 1 wherein the nodes participating in the connections or connection attempts are desktop computers, servers, embedded systems, laptop computers or a combination thereof.

8. The system of claim 1 wherein the data-packet-network is an Ethernet network connected to the Internet network and the first software application is an Ethernet driver set to operate in promiscuous mode.

9. The system of claim 1 wherein the data about the connections or connection attempts includes one, more, or a combination of sender and receiver Internet Protocol addresses; Universal Resource Locators; source

and destination ports; Transfer Control Protocol packet sequence numbers; Ethernet machine addresses; domain names; and packet header details.

10. The system of claim 1 wherein the data store comprises segregated datasets representing one or more of banned Internet Protocol addresses; banned domain names; banned Universal Resource Locators; banned network ports; and virus signatures.

11. The system of claim 1 wherein the data store further includes Ethernet machine addresses associated with bitmap icons representing individual machine types.

12. The system of claim 10 wherein certain ones of the segregated datasets are built during runtime, maintained temporarily, and searchable by one of hash table indices or binary tree indices.

13. The system of claim 10 wherein certain ones of the segregated datasets are uploaded into host Random Access Memory upon booting of the host system.

14. The system of claim 1 further including a third software application for detecting virus activity comprising:

a software routine for hashing data passed over a formed data connection; and

a software routine for comparing the hash data to a dataset containing virus signatures, the dataset searchable by hash table index, the hash entries therein derived individually from separate virus signatures.

15. The system of claim 14 wherein the hashing routine utilizes at least one sliding checksum window processing data and in the case of more than one, operating simultaneously on the data creating hash values to compare against hash entries in the hash index.

16. The system of claim 15 wherein upon detecting a hit for a virus signature, the second software application interrupts data stream processing of one or more end points of the connection by sending a reset packet to stop download of the detected virus.

17. A software application for manipulating one or more connection ends of a data network connection between two or more network nodes operating on a data-packet-network in response to detection of a pre-defined and undesirable state or states associated with the connection comprising:

a first portion thereof for detecting one or more states associated with the connection;

a second portion thereof for generating packets emulating packet activity of the connection; and

a third portion thereof for sending the emulated packet or packets to one or more parties of the connection;

characterized in that the application uses a software communication stack to send one or more Transfer Control Protocol packets emulating in construction and sequence number a packet or packets sent by a sender end of the connection, the packet received by the receiver of the connection wherein the receiving end acknowledges the packet or packets as being a valid packet or packets received from the sender of the connection, the packet or packets sent causing pre-emption or resolution of the detected state or states.

18. The software application of claim 17 wherein the data-packet-network comprises a local-area-network enhanced with Transfer Control Protocol over Internet Protocol and User-Datagram Protocol over Internet Protocol.

19. The software application of claim 18 wherein the Local Area Network is an Ethernet network connected to an Internet network.

20. The software application of claim 17 wherein manipulation of connection ends is performed by generation of and insertion of data packets to one or more nodes of the connection using Transfer Control Protocol over Internet Protocol, the generated packets emulating sender packets in construction and sequence number.

21. The software application of claim 17 wherein the packets inserted into a connection data stream are one or a combination of Transfer Control Protocol reset packets or Transfer Control Protocol FIN packets emulating at least one sending party of the connection.

22. The software application of claim 17 wherein the software communication stack is an on-board Transfer Control Protocol over Internet Protocol communication stack.

23. The software application of claim 17 wherein a pre-defined state includes one, more, or a combination of a banned Internet Protocol address; a banned Universal Resource Locator; a banned domain name; a detected virus signature; a banned port; and banned data content defined by filter.

24. The software application of claim 17 wherein the connection end nodes are desktop computers, servers, embedded systems, laptop computers, or a combination thereof.

25. The software application of claim 17 wherein Transfer Control Protocol packets are generated and inserted according to pre-defined trigger events associated with existing states or knowledge of imminence thereof discovered during operation.

26. The software application of claim 17 further including a portion thereof integrated with the first portion for detecting virus activity comprising:

a routine for hashing data passed over a formed data connection; and

a routine for comparing the hash data to a dataset containing virus signatures, the dataset searchable by hash table index, the hash entries derived individually from the virus signatures.

27. The system of claim 23 wherein the predefined state is banned content and resolution thereof includes inserting content including machine readable script by one or a sequence of TCP packets containing replacement content.

28. The software application of claim 26 wherein virus searching is supported by algorithm supporting generation and then comparison of created hash values derived from active connection data streams to hash table entries stored in a data store and to return a hit upon obtaining a match.

29. The software application of claim 26 wherein the third portion thereof is integrated with a messaging client for generating automated alerts to end nodes whose connections have been manipulated.

30. The software application of claim 26 including one or more sliding checksum windows for hashing data transferred over an active connection.

31. The software application of claim 30 wherein each checksum window processes 9 bytes of data 3-bytes at a time, each three-byte section treated as a single 24-bit number.

32. The software application of claim 26 wherein the hash table is sparsely populated and wherein the index thereof is bit-masked to reduce the overall size of the table and increase performance of the search.

33. A fast pattern search system for detecting virus patterns over a data network comprising:

a promiscuous mode driver for intercepting data packets on the network;

a hashing module for creating hash values from same-lengths of intercepted data;

a data buffer section for storing hash values; and

a processing component for comparing created hash values to an index of hash entries maintained in a data store;

characterized in that the hash entries in the data store point to virus patterns also stored in the data store and where upon a match between a created hash and a hash entry results in generation of one or more packets emulating at least one party node to the connection, the packet or packets sent to pre-empt the download of the particular virus found.

34. The system of claim 33 wherein the network is a local area network enhanced with Transfer Control Protocol over Internet Protocol and User Datagram Protocol over Internet Protocol, the Local Area Network connected to the Internet network.

35. The system of claim 34 wherein the promiscuous mode driver is an Ethernet driver and the network protected is an Ethernet network or a segment thereof.

36. The system of claim 33 wherein the length of data hashed to form a single hash value from a connection data stream is 9-bytes.

37. The system of claim 33 wherein the hashing module employs one or more sliding checksum windows and re-calculates new hash values based on data units entering or exiting the window.

38. The system of claim 37 wherein the 3-byte sections are treated as a single 24-bit number.

39. The system of claim 33 wherein the data buffer is RAM buffer.

40. The system of claim 33 wherein more than one packet is generated and sent upon a match, the packets comprising a TCP reset packet sent to the source node of the virus and a TCP FIN packet sent to the receiving node of the virus.

41. The system of claim 33 wherein more than one packet is generated and sent upon a match, the packets comprising a TCP reset packet sent to the

source node of the virus and a TCP reset packet sent to the receiving node of the virus.

42. The system of claim 33 further comprising a routine for calling a messaging client to generate a message alert and then sending the alert to the receiving node of the virus, the alert informing of the activity and providing further instruction.

43. The system of claim 33 further comprising a routine for calling a machine-to-machine messaging protocol to send a machine readable command to an application running on the receiving node of the virus, the application adapted to clean history files and any logical or physical links or references to the virus source.

44. A method for denying a connection to a data source on a data network, the connection initiated from a local network node comprising steps of:

(a) maintaining data identifying the banned data source;

(b) detecting a SYN packet from the local node sent to the host node of the banned data source, the SYN packet identifying at least the banned data source;

(c) generating a TCP reset packet emulating one sent from the local node and sending the packet to the host node of the banned data source terminating the handshake process for accessing the data source at the host node of the banned data source; and

(d) generating a TCP reset packet emulating one sent from the host node of the banned data source and sending the packet to the local node terminating the handshake process for accessing the banned data source at the local node.

45. The method of claim 44 wherein in step (a) the banned data source is identified by one or a combination of IP address, Universal Resource Locator, port identification or any portion thereof.

46. The method of claim 44 wherein in step (b) the local node is connected to an Ethernet network and the host node is maintained on the Internet network, the method of detection comprising promiscuous mode monitoring and comparison against stored data.

47. A method for stopping a download of a pop-up advertisement over a data network from a data source to a local node on the network comprising steps of:

(a) monitoring a browser session between the local node and the source node;

(b) detecting execution by the local browser of an embedded code calling an advertisement to be served;

(c) generating a TCP FIN packet emulating one sent from the data source node and sending the packet to the local node, the packet indicative that the source node has finished transmitting the ad data; and

(d) generating a TCP reset packet emulating one sent from the local node to the TCP connection source of the ad data requesting a reset of the connection preventing the source node from serving the ad data.

48. The method of claim 47 wherein the local node is connected to an Ethernet network and the data source is maintained by a node on the Internet network, the method of detection comprising promiscuous mode monitoring and comparison against stored data.

49. The method of claim 47 wherein in step (c) the FIN packet includes a machine-readable code containing one or more instruction codes for the browser application.

50. The method of claim 49 wherein in step (c) the machine-readable code is JavaScript instructing the browser not to open a container window for the ad data and to close the container window if already called.

51. A method for configuring a resource on a local network for access from the network by a node using Domain Name Service protocol comprising steps of:

(a) pre-assigning a name to the shared resource;

(b) storing the pre-assigned name in a data store;

(c) publishing the pre-assigned name to local nodes on the network;

(d) monitoring Domain Name Service requests from the local nodes;

(e) detecting the pre-assigned name in a request;

(f) generating a Domain Name Service reply emulating in construction and sequence number a reply sent from a Domain Name Server, the reply containing an IP address through which the resource may be accessed; and

(g) sending the reply to the local node that initiated the request.

52. The method of claim 51 wherein in step (a) the shared resource is one of a printer, a server node, or a network-based software application.

53. The method of claim 51 wherein in step (a) the pre-assigned name is not registered at a Domain Name Server.

54. The method of claim 51 wherein in step (d) monitoring the network for Domain Name Service requests is performed in promiscuous mode using an Ethernet driver wherein the local network is an Ethernet network.

5

55. The method of claim 51 wherein in step (e) detection is accomplished by comparing all requests for Domain Name Services made from local nodes to the store containing the pre-assigned Domain Name.

10

15